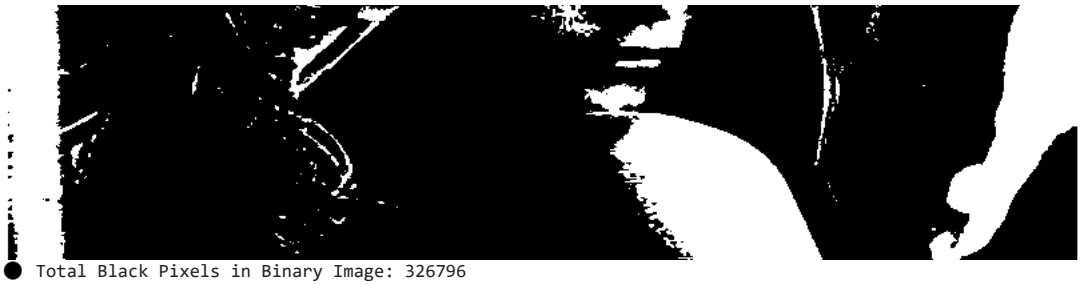
```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow  # For displaying images in Google Colab

# Load the image using the provided path
image_path = "/content/Screenshot 2025-01-30 114305.png"
image = cv2.imread(image_path)

# Check if the image is loaded correctly
if image is None:
    print(f"❌ Error: Could not load image from {image_path}")
else:
    print("✅ Image Loaded Successfully!")

    # Display the original image
    print("Original Image:")
    cv2_imshow(image)

    # Extract image size
    height, width, channels = image.shape
    print(f"📏 Image Size: Width={width}, Height={height}, Channels={channels}")

    # Calculate total number of pixels
    total_pixels = height * width
    print(f"🔢 Total Pixels in Image: {total_pixels}")

    # Convert BGR to RGB (since OpenCV loads in BGR format)
    rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    cv2.imwrite("/content/RGB_Image.png", rgb_image)
    print("✅ RGB Image Saved as 'RGB_Image.png'")

    # Convert RGB to Grayscale
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    cv2.imwrite("/content/Grayscale_Image.png", gray_image)
    print("✅ Grayscale Image Saved as 'Grayscale_Image.png'")

    # Display Grayscale Image
    print("Grayscale Image:")
    cv2_imshow(gray_image)

    # Convert Grayscale to Binary using Thresholding
    threshold_value = 127  # Adjust this value if needed
    _, binary_image = cv2.threshold(gray_image, threshold_value, 255, cv2.THRESH_BINARY)
    cv2.imwrite("/content/Binary_Image.png", binary_image)
    print("✅ Binary Image Saved as 'Binary_Image.png'")

    # Display Binary Image
    print("Binary Image:")
    cv2_imshow(binary_image)

    # Count the number of black pixels in the binary image
    black_pixels = np.sum(binary_image == 0)
    print(f"⚫ Total Black Pixels in Binary Image: {black_pixels}")
```

```
✅ Image Loaded Successfully!
Original Image:
```

```
📏 Image Size: Width=809, Height=533, Channels=3
🔢 Total Pixels in Image: 431197
✅ RGB Image Saved as 'RGB_Image.png'
✅ Grayscale Image Saved as 'Grayscale_Image.png'
Grayscale Image:
```

```
✅ Binary Image Saved as 'Binary_Image.png'
Binary Image:
```



```
Total Black Pixels in Binary Image: 326796
```

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from scipy import ndimage
from skimage import filters
from google.colab.patches import cv2_imshow  # Use this in Google Colab

# Load the image
image_path = "/content/Screenshot 2025-01-30 114305.png"
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)  # Load as grayscale

if image is None:
    print(f"❌ Error: Could not load image from {image_path}")
    exit()

print("✅ Image Loaded Successfully!")

# Display the Original Image
cv2_imshow(image)

# 1️⃣ Edge Detection Techniques
# ------------------------------

# Sobel Operator
sobel_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)
sobel_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)
sobel = cv2.magnitude(sobel_x, sobel_y)

# Prewitt Operator (Using scipy filters)
prewitt_x = filters.prewitt_h(image)
prewitt_y = filters.prewitt_v(image)
prewitt = np.hypot(prewitt_x, prewitt_y)

# Roberts Cross Operator
roberts_x = ndimage.sobel(image, axis=0)
roberts_y = ndimage.sobel(image, axis=1)
roberts = np.hypot(roberts_x, roberts_y)

# Canny Edge Detector
canny = cv2.Canny(image, 100, 200)

# Display Edge Detection Results
cv2_imshow(np.uint8(sobel))
cv2_imshow(np.uint8(prewitt))
cv2_imshow(np.uint8(roberts))
cv2_imshow(canny)

# Save Edge Detection Outputs
cv2.imwrite("Sobel_Edge.png", np.uint8(sobel))
cv2.imwrite("Prewitt_Edge.png", np.uint8(prewitt))
cv2.imwrite("Roberts_Edge.png", np.uint8(roberts))
cv2.imwrite("Canny_Edge.png", canny)

# 2️⃣ Image Segmentation Techniques
# ---------------------------------

# (i) Global Thresholding
_, global_thresh = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY)
cv2_imshow(global_thresh)
cv2.imwrite("Global_Threshold.png", global_thresh)

# (ii) Adaptive Thresholding
adaptive_thresh = cv2.adaptiveThreshold(image, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                        cv2.THRESH_BINARY, 11, 2)
cv2_imshow(adaptive_thresh)
cv2.imwrite("Adaptive_Threshold.png", adaptive_thresh)

# (iii) Edge Detection for Segmentation (Using Canny)
cv2_imshow(canny)
```